Hierarchical Pattern Mapping

Cyril Soler
Marie-Paule Cani
Alexis Angelidis

iMAGIS - GRAVIR/IMAG - INRIA

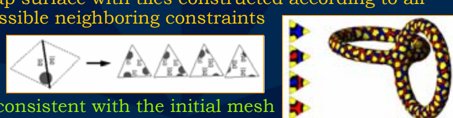
Motivation

- Seamlessly texture a mesh using a texture sample

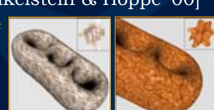
- Difficult because
  - Generally no continuous parameterization of the mesh
  - It's hard to texture locally without deformations
  - Very few information in the input sample

Previous works - Pattern mapping

- Pattern-based texturing [Neyret & Cani '99]
  - Map surface with tiles constructed according to all possible neighboring constraints

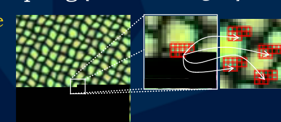


  - Inconsistent with the initial mesh
- Lapped textures [Praun, Finkelstein & Hoppe '00]
  - Paste pre-cut tiles on surface
  - Blend borders at rendering
  - Needs a specific rendering algorithm or extra texture storage



Previous works - On-mesh synthesis

- Non parametric sampling [Efros & Leung'99]
  - Use pixel coherence



- 3D Point-based synthesis [Turk'2001, Wei & levoy 2001]
  - Proceed hierarchically
  - Produces a collection of colored points in 3D
  - Needs a specific rendering algorithm or extra texture storage

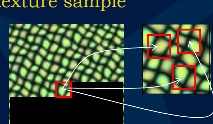
Previous works - Conclusion

- No 3D method provides at the same time
  - Initial mesh conservation
  - Initial texture sample conservation

⇨ This is what we would like to do

- Related work in 2D: [Efros & freeman 2001]
  - Paste blocks selected from texture sample
  - Reduce discontinuities

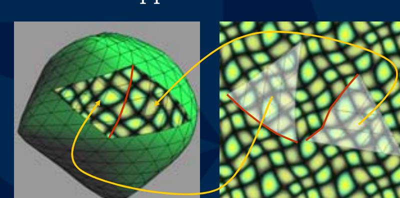
B B ⇨ B B



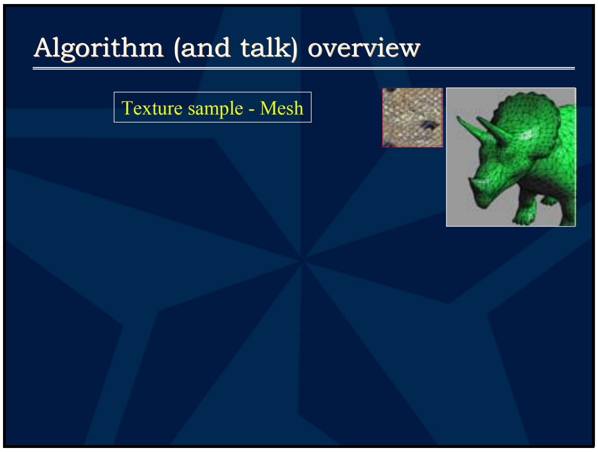
  - Is it possible on a mesh?

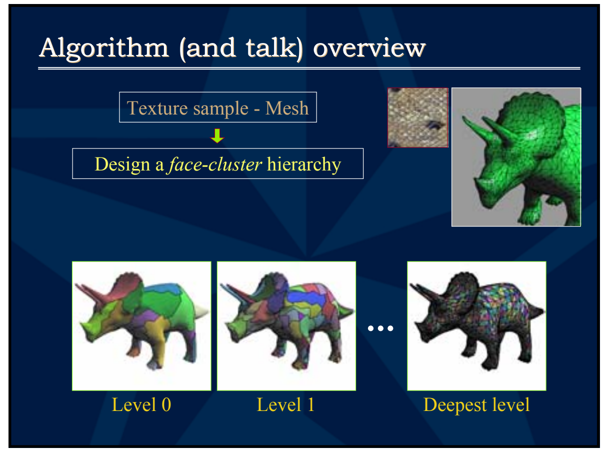
Proposed approach

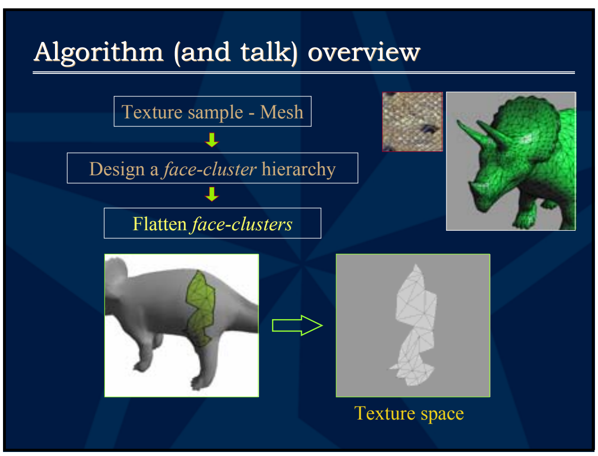
- Select independent regions in the texture that match once mapped on the mesh

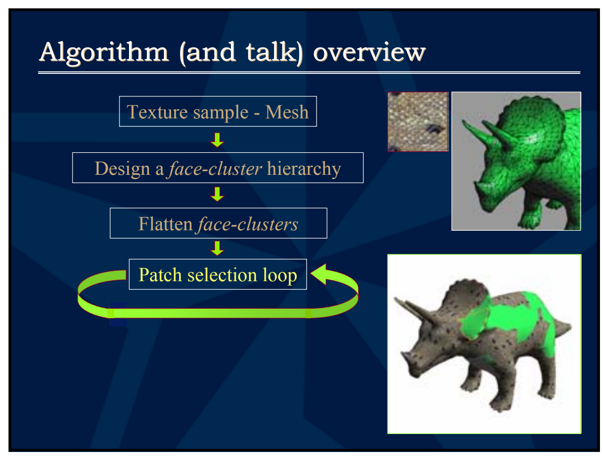


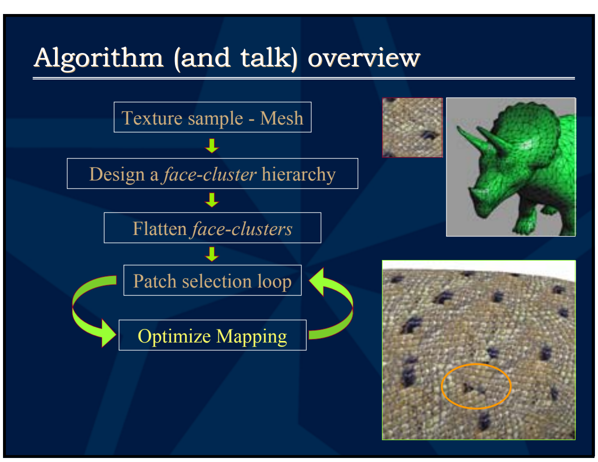
- Advantages
  - Original mesh and texture sample are preserved
  - Computed data (*e.g. texture coordinates*) is portable

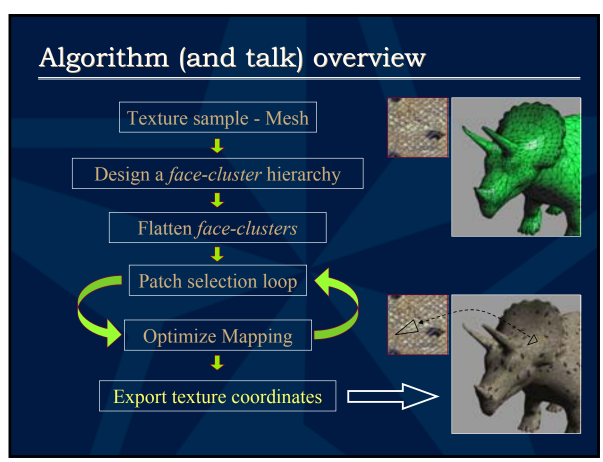

Algorithm (and talk) overview
Texture sample - Mesh


Algorithm (and talk) overview
Texture sample - Mesh
Design a *face-cluster* hierarchy
Level 0
Level 1
…
Deepest level


Algorithm (and talk) overview
Texture sample - Mesh
Design a *face-cluster* hierarchy
Flatten *face-clusters*
Texture space

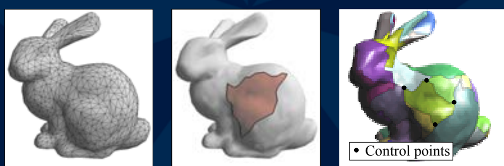

Algorithm (and talk) overview
Texture sample - Mesh
Design a *face-cluster* hierarchy
Flatten *face-clusters*
Patch selection loop


Algorithm (and talk) overview
Texture sample - Mesh
Design a *face-cluster* hierarchy
Flatten *face-clusters*
Patch selection loop
Optimize Mapping


Algorithm (and talk) overview
Texture sample - Mesh
Design a *face-cluster* hierarchy
Flatten *face-clusters*
Patch selection loop
Optimize Mapping
Export texture coordinates

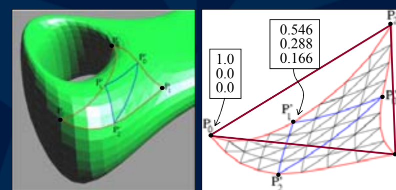
Design a *Face-Cluster* hierarchy

- Requirements:
  - Face-clusters should be able to project on a plane
- Simple subdivision method:
  - Start with n seed faces (randomly chosen)
  - Assign mesh faces to the sub-cluster of closest seed



Flattening face-clusters

- For each face-cluster in texture space
  - Pre-compute relative position of control points w.r.t. parent control points in texture space
  - Use barycentric coordinates
  - Compute them with a heuristic

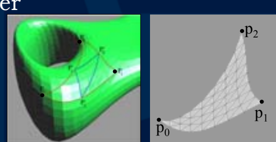


Surface space | Texture space

Flattening face-clusters

- To flatten a face-cluster
  - Position parent control points
  - Recursively compute point positions
- Advantage
  - Real-time update when control points move

  ⟹ Useful to optimize fitting



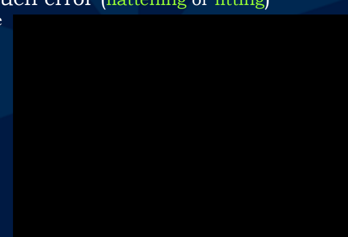
Face-cluster selection algorithm

Select and texture face-clusters until total coverage

Rules:

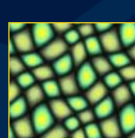
1. Select clusters at highest possible level
2. Propagate mapping to neighboring clusters
3. If too much error (flattening or fitting)

   Subdivide



Texture patch fitting

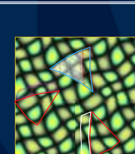
- Extraction of a mask

- Fitting problem
  - Can we find  somewhere into  ?

↳ gives a possible position for the neighbor

Texture patch fitting

- Example solutions:



- Best match searching for a translation x
  - Minimize L_2 distance between I and T over J

$$E(x) = \sum_y J(y)\,(I(y) - T(x+y))^2$$

I 

- Direct computation is costly !!

T  *J* 

Texture patch fitting

- Express $E(x)$ using image correlation

$$E(x) = \sum_y I(y)^2 + (-2\ I \circ T + J \circ (T^2))\ (x)$$

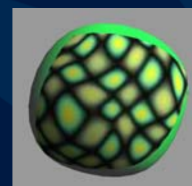
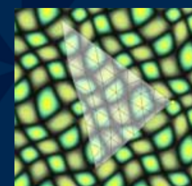
- Compute correlation using FFT ($\mathcal{F}$)

$$f \circ g = \mathcal{F}^{-1}(\mathcal{F}(f)\overline{\mathcal{F}(g)})$$

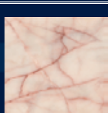
  - Only *F(I)* and *F(J)* must be re-computed at each search
  - *F(T)* and *F(T^2)* are computed once and saved.
- Pre-compute *F(T)*, *F(T^2)* for various orientations
- Sample topology is not necessarily toroidal

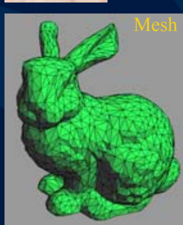
Mapping optimization

- For each newly mapped face cluster
  - Minimize discontinuity along edges with neighbors
  - Recursively moving control points

Results - (1) isotropic pattern


Sample


Mesh


Result (Pentium III)
2 mn

Results - (1) isotropic pattern


Sample

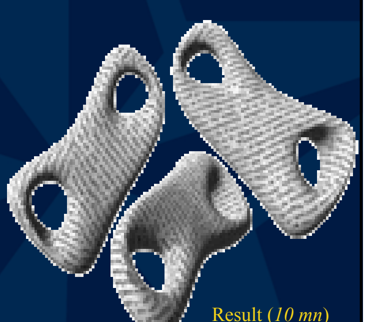

Mapping


Result
(*21 mn*)

Results - (2) anisotropic pattern


Sample

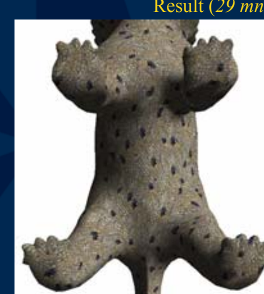

Mapping


Result (*10 mn*)

Results - (3) fun


Sample


Mesh


Result (*29 mn*)

Results - (3) fun

Samples

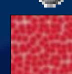
Conclusion - Future work

- Advantages
  - Preserves initial texture sample and mesh geometry
  - Exports texture coordinates only
- Limitations
  - Mesh resolution should be finer than texture features
  - The mapping is (almost) never perfect

⇨ Still consistent with input mesh resolution
Trade-off: enable local mesh refinement

- Improvements
  - Allow human intervention during algorithm
  - A better clustering would increase speed

Thanks
for listening



Design a *Face-Cluster* hierarchy

- Few requirements
  - Face-clusters should be able to project on a plane

⇨ no need for complex methods

- General meshes: simple subdivision method:
  - Start with n seed faces (randomly chosen)
  - Assign mesh faces to the sub-cluster of closest seed
- Subdivision surfaces: intrinsic subdivision